Radiation Heat Transfer

Final Project Report

By Mikhail Lara

Review of Numerical Methods

Radiative transfer through a medium is governed by four phenomena which determine the degree of extinction and augmentation of radiative intensity. These four phenomena are radiation absorption, out-scattering of intensity, volumetric emission, and in-scattering of intensity. The first two phenomena contribute to extinction of radiation intensity by converting it into internal energy and redirecting photons away from a given incident direction. The latter two augment radiation transfer by local blackbody emission and by in-scattering, which directs photons out-scattered from other locations into the local direction of interest. These competing augmentative and extinctive effects are collectively described by the radiative transfer equation(RTE). In general, the RTE is defined in terms of both variation of intensity in both time and space but the contribution of the time derivative is typically neglected because of the coefficient of c_o^{-1}, converting the RTE from a partial differential form to an ordinary differential form. Even with this simplification, the RTE is a difficult equation to solve analytically because of the effect of in-scattering. Unlike classical ordinary differential equations, the equation of radiative transfer is an integro-differential equation because it incorporates the integration of intensity from all incoming directions. This integration implicitly couples intensity at a given point with the intensity from all locations in a medium because of the effect of in-scattering. It is for this reason that numerical techniques are commonly used to study radiative transfer in scattering media.

There are three numerical techniques that are commonly implemented to simulate radiative transfer. What distinguishes these numerical methods from each other is the manner by which they approximate the integral corresponding to in-scattering. The technique most frequently implemented in commercial radiation solvers is the method of discrete ordinates(S_n). This method is the most computationally demanding because handles the integration by what is essentially a brute force method. It resolves the effect of in-scattering by discretizing the integration over the entire bounding unit sphere into a finite number of solid angles over which incident intensity is assumed to be uniform with the contribution from each solid angle determined by empirically determined weightings. As should be expected, the numerical treatment of the integral approaches an analytical treatment of the RTE for sufficiently high solid angular discretizations. In commercial solvers such as ANSYS FLUENT and COMSOL, the bounding unit sphere is discretized into eight distinct solid angles. This discretization results in a set of eight coupled ordinary differential equations corresponding to radiative transfer of intensity for each of the discretized directions. The set of coupled differential equations can be readily solved using iterative techniques.

A slightly simpler numerical technique used to solve the RTE is the method of spherical harmonics. This numerical technique handles the coupling of intensity due to in-scattering by representing the intensity function and the scattering phase-function as Fourier series in order to take advantage of the orthogonally of the basis functions, specifically Legendre polynomials. The advantage of this representation of the RTE is that integration of orthogonal basis functions over the specified period results in a zero sum, reducing the problem to a set of linear algebraic equations with the unknown variables being the Fourier coefficients. Since the resultant equations are linear, they can be readily solved using existing matrix elimination techniques. The main disadvantage of this technique is that the Fourier series must to be carried to a high number of terms in order to be accurate for arbitrarily complex systems. Experience shows that low order approximations for the intensity and scattering phase function will yield poor results unless the system inherently exhibits spherical symmetry. In addition, the accuracy of this method increases very slowly with increasing number of harmonics.

The numerical technique that is the simplest to implement is the Schuster-Schwarzchild Approximation. This is the technique used for the in-house simulations of this study. This numerical approximation for the RTE is the most simplistic because it assumes that the radiative intensity in the forward and backward directions can be treated as constants. By assuming this, the integro-differential

form of the RTE is reduced to a set of two linear ordinary differential equations that are coupled by the in-scattering terms due to upper hemispherical and lower hemispherical intensity. This numerical method is equivalent to a discrete ordinates representation of the RTE where the bounding unit sphere is discretized into two hemispheres of constant intensity. Generally speaking, the Schuster-Schwarzchild Approximation, also known as the two-flux approximation, is only valid for systems that exhibit plane symmetry with a small optical thickness. These restrict the applicability of this approximation to systems that can be modeled as being strongly one-dimensional with negligible attenuation of intensity.

The formulation of the RTE used in this study is the general Schuster-Schwartchild Approximation given by Modest[1]. To simplify the integral in the formal definition of the RTE, the intensity function is assumed to be constant in the upper and lower hemispheres of the bounding unit sphere. This assumption simplifies the integration over the upper and lower hemispheres and simplifies the RTE to a set of two coupled equations, shown by Equations 1 and 2**,** that describe radiative transfer of the intensities in the upper and lower hemispheres.

$$\frac{dI^+}{dx} = 2\kappa I_b - 2\beta I^+ + \frac{\sigma}{\pi}(I^+ + I^-) \qquad \text{(Eq.1)}$$

$$\frac{dI^-}{dx} = -2\kappa I_b + 2\beta I^+ - \frac{\sigma}{\pi}(I^+ + I^-) \qquad \text{(Eq.2)}$$

These coupled equations are solved only on the nodes located between the two boundaries using a second-order fully explicit scheme for Case 1 and a first-order iterative explicit Euler scheme for Cases 2 and 3. Since an explicit Euler scheme requires knowledge of the solution at the previous location, the boundary conditions were used to initialize the relevant intensity at the nodes immediately adjacent to the walls. Using these initial values for intensity, the explicit Euler scheme calculated values for I^+. Because the Schuster-Schwarzchild equations are defined with respect to specific forward and backward directions, the $\mathbf{I}^+$ and $\mathbf{I}^-$ transfer equations are solved in opposite directions using the estimates for intensity from the previous iteration as inputs for the derivative terms of the current iteration. The solutions were defined as having converged when the maximum relative error at any node between successive iterations dropped below 0.001. In order to preserve the numerical values of the boundary conditions, only the intensities at the internal nodes are modified during the iteration loop.

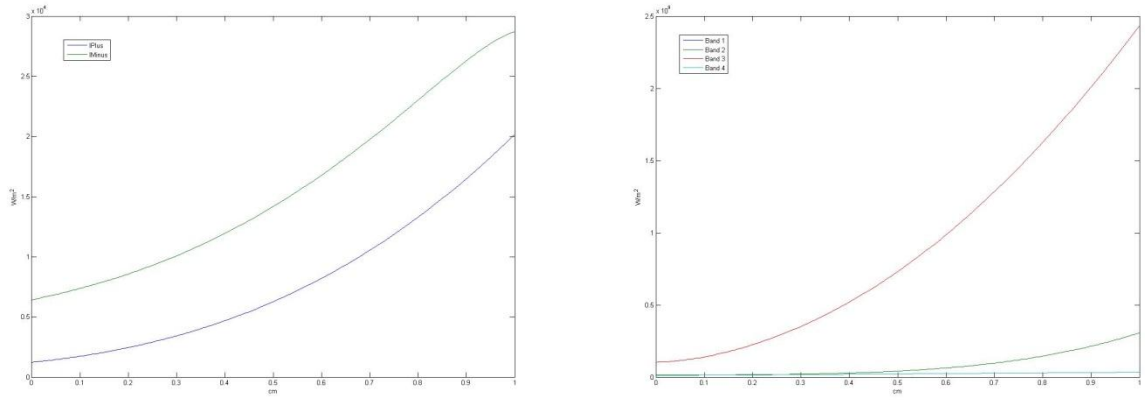
Comparison of the accuracy and speed of the Schuster-Schwarzchild solver to that of commercial solvers was evaluated by comparing wall heat flux values from the in-house solvers to those generated by the discrete ordinates method implemented in ANSYS FLUENT. Heat flux was used instead of intensity because FLUENT does not provide access to intensities calculated by the discrete ordinates method. In addition, incident intensity was intentionally not evaluated because it does not provide information about the distribution of intensity at a given point. A user-defined function was not required for discrete ordinate simulations of radiative transfer in gray, absorptive, isotropically scattering media because these conditions are the default settings in FLUENT. However, one was used in order to incorporate the effects of an anisotropic scattering phase function in Case3. The thermal conductivity of the computational domain was specified to an artificially high value of 10^{15} W/(m^2·K) to guarantee that the steady state temperature distribution across the domain remained linear. This steady state temperature profile was verified for all simulations. The temperature profile and a drop in residuals below 10^{-6} were the criteria for defining a simulation as fully converged.

Validation of the Schuster-Schwarzchild in-house code was performed by comparing profiles of heat flux divergence for optical lengths of 1 and 2.5 to analytical distributions reported by Xia[1] for non-scattering media. For each case, the in-house solver generated results that were similar to published distributions. The similarity between simulated and reported distributions could not be established for higher values for optical length. The mismatch at higheer values of optical thickness is

expected because the Schuster-Schwarzchild Approximation assumes an optically thin medium and Xia reports that the approximation begins to break down at an optical length of 5.

Case 1

The Schuster-Schwarzchild Approximation for radiative transfer through a medium which is non-scattering with a spectrally dependent absorption coefficient predicts that the total intensities in both the upper and lower hemispherical directions increase from the cold wall to the hot wall, as shown in Figure 1A. This trend is expected because blackbody intensity, which increases with temperature for all wavelengths, contributes to radiation augmentation in the RTE. Since the temperature across the slab increases linearly position zero to the position of 1cm, augmentation by blackbody intensity becomes more dominant as the distance from the cold wall increases. However, this argument does not apply for all wavelengths.



Figures 1a & 1b: Total intensities across slab(left). Spectral intensities for each band(right).

In the specified temperature range of 500K to 1000K, the spectral contribution of blackbody intensity to radiation augmentation is significant only in the spectral range corresponding to wavelengths greater than 1μm. This is a direct result of the bivariant dependence of the Planck distribution on both temperature and wavelength. Since blackbody emissive power at 1000K decreases rapidly for wavelengths below 1μm, high frequency radiation in the ultraviolet domain does not contribute significantly to radiative transfer in the system of interest and can be neglected with regards to discussions of total intensity and total heat flux. Inspection of the results for upper hemispherical intensity(I^+) and lower hemispherical intensity(I^-) generated by the Schuster-Schwarzchild in-house code support this simplification because they report maximum intensity values of 10^{-9} W/m^2 for the first spectral band. The dominant spectral range for the simulated system is $\lambda > 0.5\mu m$ where the magnitudes of I^+ and I^- reach 10^7 W/ m^2. A sample of I^- profiles from each spectral band is illustrated in Figure 1B.

An interesting characteristic of the total intensity profiles predicted by the Schuster-Schwarzchild in-house code is the change in concavity of the I- profile in the region near the hot wall. The I^- and I^+ intensity results show that both profiles demonstrate upward concavity from the cold wall to approximately 60mm across the slab indicating that the derivative of intensity is increasing everywhere in that region. However when the distance from the cold wall increases beyond this range 60mm, the I^- curve exhibits a change to a downward concavity. This change in shape indicates that in the region near the hot wall, the dynamics of the RTE shift such that extinction by absorption dominates on the total basis.

The distribution of total heat flux across the slab, calculated from numerical integration of all spectral intensity data, is presented in Figure 2. The numerical integration scheme used to calculate the

distribution was the trapezoidal rule. Everywhere in the slab the heat flux is directed from the hot wall to the cold wall, which is indicated by a values less than zero. The direction and decrease in magnitude of heat flux agree with physical expectations because the volumetric temperature in the slab decreases from the 1cm position to the cold wall located at 0cm with the temperatures at the boundaries being continuous with the volumetric temperature profile.

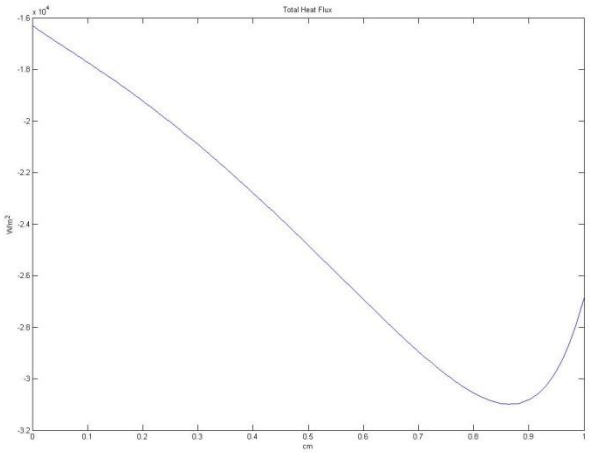


Figure 2: Total heat flux across slab.

An interesting feature of the heat flux profile is the noticeable peak located approximately 15mm from the hot wall. This effect must be due to the temperature dependence of radiation augmentation because in the case of a non-scattering medium, the only augmentation term is due to blackbody emission. This peak cannot be due to temperature at the boundary because that would place the peak exactly at the boundary where the temperature is at its maximum.

The radiative heat fluxes predicted by the in-house code and those predicted by the discrete ordinates method implemented by ANSYS FLUENT indicate that for the spectral variation in absorption coefficient specified for Case 1, the Schuster-Schwarzchild Approximation for the RTE is unable to generate results that are comparable to the commercial standard. The in-house simulations over-predict wall heat fluxes by as much as factor of 130 greater than those predicted by FLUENT, reporting heat fluxes of 16292 W/m^2 and 26842 W/m^2 at the cold and hot walls respectively. These values are significantly greater than the values of 124 W/m^2 and 953 W/m^2 calculated by FLUENT. The discrepancy between the results is most likely due to the combined effects of the absorption coefficients associated with Band 3 and Band 4. The simplification of the RTE to constant intensities in the upper and lower hemispheres is most accurate for optically thin media. When the optical thickness becomes large, the validity of the simplification breaks down leading to erroneous values for intensity. Xia[1] reports that beyond an optical thickness of 5, the Schuster-Schwarzchild Approximation for the RTE breaks down and fails to accurately predict the divergence of heat flux. The optical thicknesses simulated in Band 3 and Band 4 are 6 and 100 respectively. Since the simulated optical thicknesses are greater than the critical value by as much as two orders of magnitude, the poor agreement between the discrete ordinates results and the Schuster-Schwarzchild solver is likely due to inadequacy of the uniform hemispherical simplification to model radiation transfer in optically thick media. The values reported here correspond to solutions that are mesh-independent, defined by a drop in relative errors below 5% between successive meshes.

Case 2

The system under investigation in Case 2 involves radiative transfer in a gray, absorptive,

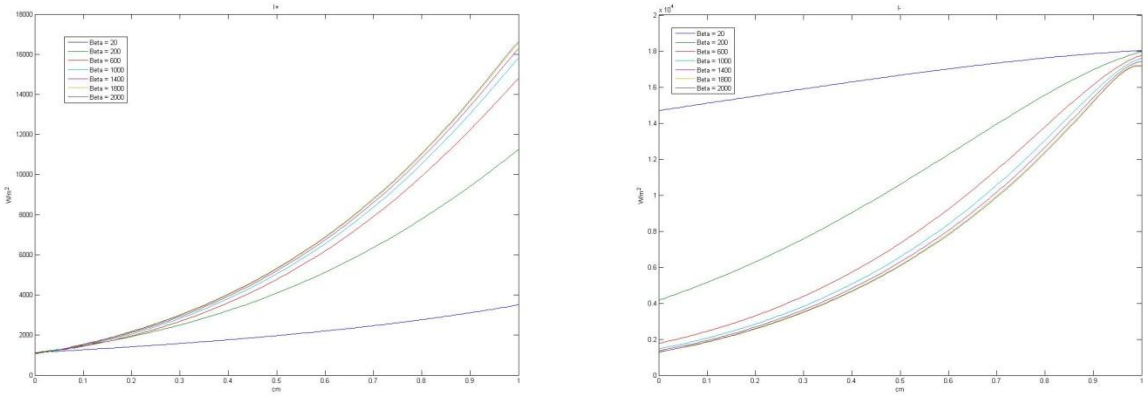
isotropically scattering medium. The dynamics simulated in this case are distinct from those simulated in Case 1 because a non-zero value for the scattering coefficient(**σ)** is used to incorporate the effects of extinction and augmentation by out-scattering and in-scattering. Although the conditions simulated in this case are independent of spectral range, they are more complex than the previous case because of scattering. Rather than simplifying the problem to a set of uncoupled linear ordinary differential equations, radiation augmentation by in-scattering requires that the problem be solved using the general integro-differential form of the RTE. Incorporating the effects of scattering into the RTE makes predicting of the intensity distribution in the medium difficult because the directional derivative of intensity becomes dependent not only on the values of absoption coefficient(**κ)** and **σ** but also on the local intensity function integrated over the entire bounding unit sphere.

The aim of Case 2 is to characterize the dependence of intensity and radiative heat flux on κ and σ. Because these coefficients contribute to extinction and augmentation of radiation in a coupled manner, a parametric study was performed to identify the coupled and isolated effects of these coefficients on radiative transfer. These effects are characterized by running simulations in which κ and σ are varied in under three different conditions: (1) κ and σ are equal, (2) κ is varied with σ held constant, and (3) σ is varied with κ held constant. All simulated conditions are listed in Table 1. It should be noted that for Condition 2 and Condition 3, the coefficient that is held constant is assigned a value of 500 m^{-1}, which places the medium in the optically thick domain. This is done so that the simulations would demonstrate the radiative dynamics when the coefficient of interest is both less than and greater than the fixed coefficient.

	Simulated Values for κ & σ (Condition 1)										
κ = σ	10	100	200	300	400	500	600	700	800	900	1000
	Simulated Values for σ (Condition 2)										
κ = 500	10	100	200	300	400	500	600	700	800	900	1000
	Simulated Values for κ (Condition 3)										
σ = 500	10	100	200	300	400	500	600	700	800	900	1000

Table1: Simulated Conditions for Case 2.

As shown in Figure 3A and Figure 3B, the distribution of intensities in the upper and lower hemispheres increase significantly as κ and σ are varied from $10m^{-1}$ to $1000m^{-1}$, which correspond to optical lengths of 0.2 and 20 respectively. All the simulations for Condition 1 produce intensity distributions that increase monotonically from the cold wall to the hot wall. This result makes physical sense because the temperature of the slab is defined as increasing linearly from the cold wall to the hot wall in a manner such that the temperature at the walls is equal the imposed temperature boundary conditions. This increase in temperature corresponds to an increase in the blackbody intensity distribution from the cold wall to the hot wall. Since the blackbody intensity contributes to radiation augmentation in the RTE and assuming that the local intensity is much less than the blackbody intensity, the derivatives of upper hemispherical and lower hemispherical intensity are expected to increase dramatically as the distance from the cold wall increases. This effect should be especially pronounced in these simulations because the minimum and maximum temperatures correspond to blackbody intensities of 1127 $W/(m^2 \cdot K)$ and 18048 $W/(m^2 \cdot K)$. This temperature dependent behavior is clearly demonstrated by both the I^+ and I^- distributions across the slab as they exhibit an upward concavity that is greatest near the hot wall.



Figures 3A & 3B: I^+(left) and I^-(right) profiles across the slab for equal κ and σ.

For the simulation performed when κ and σ are assigned values of $10m^{-1}$, the I^+ and I^- distributions across the slab demonstrate the least increase of any of the simulations of Condition 1. The upper hemispherical intensity at the hot wall is only twice the intensity value of the cold wall whereas the intensity is 14 times the cold wall intensity when κ and σ equal $1000m^{-1}$. The effect of an optically thin medium is even more pronounced for lower hemispherical intensity, which increases by only 22% from cold wall to hot wall compared to 1236% increase for the optically thickest simulation. These results follow from the definition of extinction coefficient. Since extinction coefficient is a sum of κ and σ, the contributions of these coefficients to augmentation scale equally to that of β. This implies that in the limiting case of β approaching a value of zero, all augmentation and extinctive terms in the RTE become negligible resulting in a constant intensity value. The numerical result for I^+ and I^- when κ and σ equal $10m^{-1}$ support this scaling argument because these values of I^+ and I- are significantly lower than the values in the other simulations.

The most interesting result of this simulation is the behavior of intensity as β is increased, shifting the slab from being an optically thin medium to being an optically thick medium. For the I^+ distributions, increasing the value of β results in an increase in intensity values everywhere along the slab with the greatest increase in intensity occurring at the hot wall. The increase in intensity indicates that augmentation by emission and in-scattering dominate the I^+ equation defined in the Schuster-Schwarzchild Approximation. Dominance by augmentation is not present in the results for the I- distributions. Rather than increase in value, they uniformly decrease in magnitude as β increases as shown in Figure 3A. This indicates that unlike the I^+ equation, radiation extinction is dominant in the I^- radiative transfer equation as long as κ and σ are equal. It is important to recognized that although the dependence of I^+ and I^- on β are opposite to each other, the value of I^- is greater than the value of I^+ at every location on the slab. It is worth noting that as β is increased to a value of $2000m^{-1}$**,** the spacing between the intensity distributions decreases rapidly until they essentially lie on top of each other. This collapse of intensity profiles to a single curve suggests that as the extinction coefficient increases, the intensities across the slab asymptotically converge to an upper, or lower, limit that is independent of the specific value of β.

Distributions for radiative heat flux across the slab were calculated using the numerical solutions for I^+ and I^- and are shown in Figure 4. The directional convention adopted for these simulations is that negative heat flux is directed towards the cold wall. Therefore all heat flux values reported plotted have negative values, which is consistent with entropy balance. As expected, the simulated results for κ and σ equal to $10m^{-1}$ indicate that when the slab is optically thin, the radiative heat flux remains relatively constant.

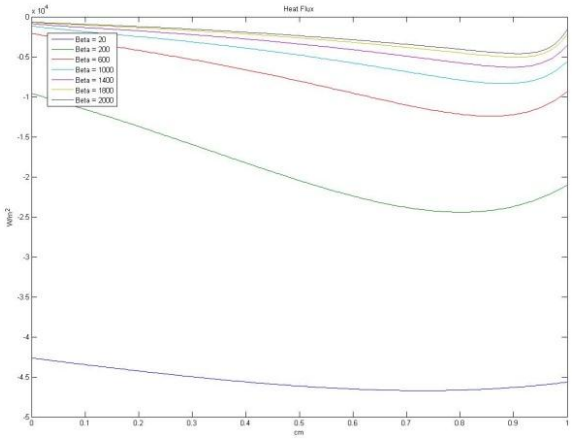
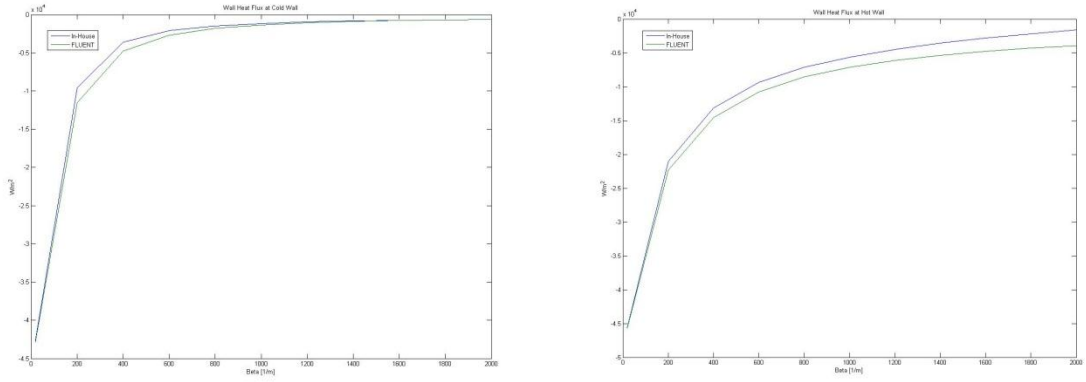


Figure 4: Heat flux profiles for the Condition 1 simulations.

A comparison of the results at the walls shows that the heat flux changes by only 6% from the hot wall to the cold wall. With regards to the effects of radiation extinction on heat flux, the distributions of heat flux decrease dramatically with increasing values of β and drop below 5000 W/m^2 when κ and σ are equal $500m^{-1}$. This result is expected because heat flux should decrease as radiation extinction becomes dominant and is supported by the nearly flat heat flux profile corresponding to κ and σ equal $1000m^{-1}$.

Comparison of the wall heat fluxes predicted by the Schuster-Schwarzchild solver to those generated for the discrete ordinates method implemented in ANSYS FLUENT show strong agreement between the results of the in-house code and those of the commercial standard with the relative errors of only 58% at the hot wall and 9% at the cold wall for the optically thickest case. As shown in Figures 5A and 5B, the heat fluxes predicted by the in-house solver are very similar to the values calculated by the commercially available discrete ordinates method.

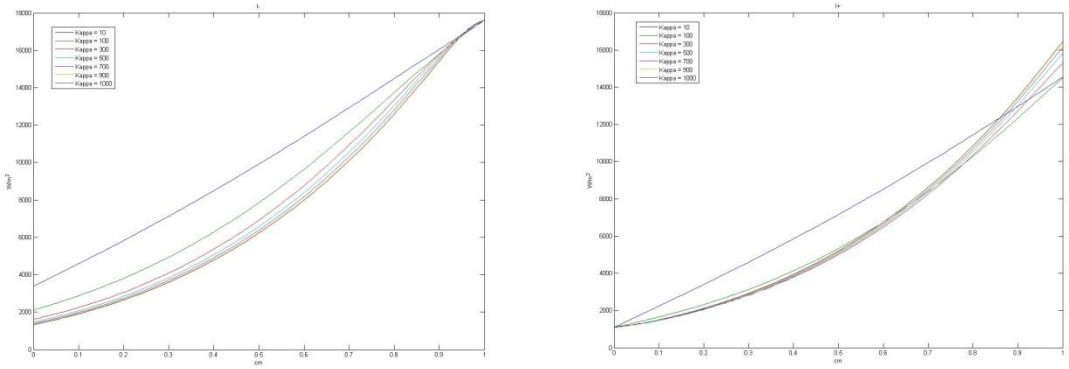
This degree of accuracy was not expected because the Schuster-Schwarzchild Approximation is formally only valid for optically thin media. Since Xia reports a breakdown of the approximation at an optical length of 5, the agreement between the discrete ordinates solver and the in-house solver was expected to fail for all cases beyond κ and σ equal $200m^{-1}$. However, the results indicates that the Schuster-Schwarzchild Approximation for plane parallel radiative transfer in gray, absorptive, isotropically scattering media is a valid simplification for media of "medium" optical thickness when κ and σ are equal.



Figures 5A(Left) and 5B(Right): Comparison of Wall Heat Fluxes predicted by in-house code and FLUENT.

The goal of the Condition 2 simulations is to characterize the dependence of intensity and heat

flux on absorption coefficient in a quasi-isolated fashion. In order to accomplish this, the value of scattering coefficient was held fixed at $500m^{-1}$ while absorption coefficient was varied from $10m^{-1}$ to $1000m^{-1}$. Distributions for upper hemispherical intensity, lower hemispherical intensity, and radiative heat flux for all simulations are presented. For the optically thinnest case when κ is equal to $10m^{-1}$, the I^+ and I^- profiles increase linearly from the cold wall to the hot wall, as shown in Figure 6A and 6B. This is a physical result because it occurs when the extinction coefficient is at its minimum value.



Figures 6A(left) & 6B(right): Intensity profiles for Condition 2.

It is surprising however, that the profile appears to be perfectly linear while maintaining a large positive slope of approximately 1344100 W/m^3 and 1420800 W/m^3 for I^+ and I^- respectively. Since these linear profiles are produced when absorption and emission effects are negligible these linear intensity profiles indicate that when a gray, plane-parallel, absorptive, isotropically scattering system is dominated by scattering, the effects of in-scattering and out-scattering sum to a constant.

In the region near the cold wall to approximately 60% the distance across the slab, all I^+ distributions collapse to a single curve for all simulations when κ is greater than $10m^{-1}$. This rapid drop in I^+ is evidence of absorptive radiation extinction dominating over radiative augmentation due to emission and in-scattering.

In the region near the hot wall, the curves separate and demonstrate that I^+ increases with increasing values for absorption coefficient. This behavior is likely a temperature dependent result because increases κ can only directly increase intensity if the local blackbody intensity is greater than upper-hemispherical intensity. This condition is most likely satisfied in the region near the hot wall because the temperature within 0.3 cm of the hot wall exceeds 850K, which means that in the minimum blackbody intensity is $9421W/m^2$ whereas the blackbody intensity at the cold wall is only $3543W/m^2$. Dominance of radiation augmentation by emission over radiation extinction is further supported by the range of intensities at the nodes immediately adjacent to the walls, where the temperature is at its maximum. Rather than converging to a single value, I^+ intensities at the nodes adjacent to the hot wall range from $14540W/m^2$ to $16461W/m^2$. In addition, a change in the dominance of augmentation over extinction near the hot wall is most clearly demonstrated by cross-over of all the intensity distributions the optically thinnest simulation when κ is equal to $10m^{-1}$. This crossing of intensity distributions indicates that in the region near the hot wall there exists a unique temperature at which integrated augmentation due to emission exceeds integrated augmentation of a non-absorbing, isotropically scattering media. The only simulation that does not exhibit this change in dominance is when κ is equal to $100m^{-1}$ where instead of crossing the distribution corresponding to κ being equal to $10m^{-1}$, it matches its value at the node adjacent to the wall.

Unlike the upper-hemispherical intensity distributions, the lower-hemispherical intensities do not immediately collapse to a single curve for κ being greater than $10m^{-1}$. As κ increases, the distributions for I^- progressively decrease and seem to approach an asymptotic lower limit. After κ is increased to $1000m^{-1}$, which corresponds to an extinction coefficient of $1500m^{-1}$, the change in the

magnitude and shape of the I^- distributions for successive values of κ becomes negligible. This trend of progressive collapse of the intensity distributions to a single curve was also observed in the simulations performed for Condition 1. It is worth noting that just as there were indications of a change in dominance between augmentation due to emission and total extinction, there is also a small region near the hot wall where the linear, optically thin profile crosses the other intensity profiles. It is important to recognize that all the simulations aimed at isolating the effects of absorption coefficient on radiative transfer were done using a value for scattering that is defines the medium as being optically thicker than the optically thinnest settings used in Condition 1. The optical thickness of the medium is the primary reason why all simulations I^+ and I^- distributions converge to the same value at the cold and hot wall, respectively.

The heat flux profile across the slab when κ is equal to $10m^{-1}$ is relatively flat and decreases in magnitude by only 26% from the hot wall to the cold wall, as compared to the 79% decrease for the simulation where κ is equal to $1000m^{-1}$. This is to be expected because this value for κ corresponds to an optical thickness of 5.1 which is still in the optically thin range as reported by Xia. The results for dependence of radiative heat flux on κ are similar to the results for dependence on β in Condition 1, the heat flux profiles across the slab decrease in magnitude as the value of κ is increased from $100m^{-1}$ to $1000m^{-1}$ as is apparent in Figure 7.

This result makes sense physically because as extinction effects become dominant, heat flux should asymptotically decrease to a profile that is dependent on only the temperatures at the boundaries and the temperature profile in the slab. The asymptotic heat flux profile is a direct consequence of the asymptotic distributions for I^+ and I^- as the absorption coefficient approaches infinity.

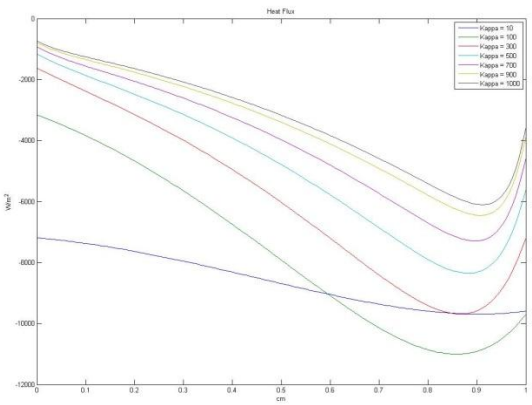
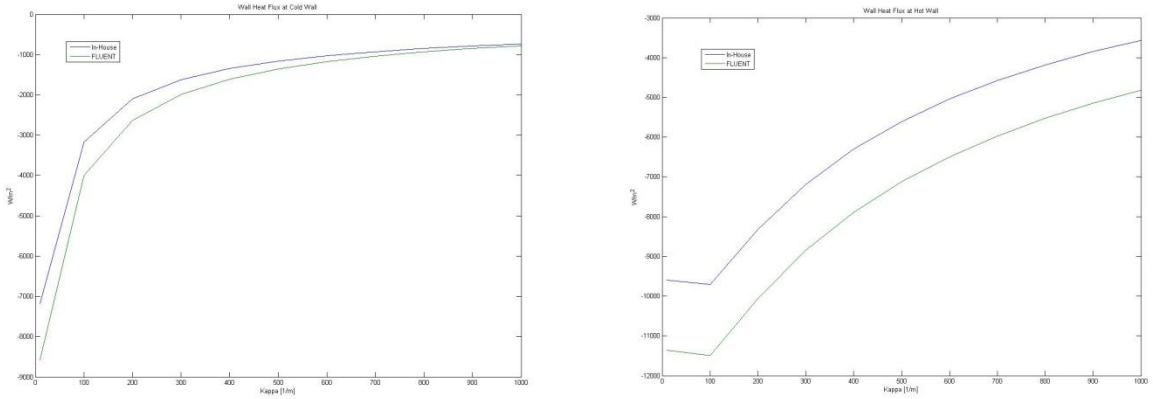


Figure 7: Heat flux profiles for Condition 2.

The most interesting features for heat flux are the cross-over of the heat flux profile for optically thin media with three optically thicker profiles and maxima in all the profiles located near the hot wall. The maxima in the heat flux profiles indicate that as long as the medium is of a sufficient optical thickness, the contribution of κ to absorption in the I^+ equation will result in a location in the region near the hot wall where heat flux is maximized. The exact location between the boundaries is dependent on the specific values of κ and σ as well as the imposed temperature conditions. However, because the optically thinnest profile is actually smaller in magnitude than the profile for κ being equal to $100[m^{-1}]$, the existence of a clear maxima is only apparent when κ is large enough that augmentation by emission is significant. This effect must be due to absorptive dominance in the I^+ equation because the results for I^- indicate that it has a weak dependence on κ that does not generate the change required to create the observed peak in the heat flux. The changes in the behavior across the slab for I^+, I^-, and heat flux provide strong evidence that radiative transfer for this system is a very strong function of κ.

Comparison of the radiative wall heat fluxes produced by the in-house code to those simulated by the discrete ordinates method in FLUENT demonstrates significant discrepancies with regards to accuracy. At both the hot wall and the cold wall, the explicit formulation of the Schuster-Schwarzchild Approximation implemented by the in-house solver consistently underestimates the value of radiative heat flux for nearly all values of κ as shown in Figures 8A and 8B.

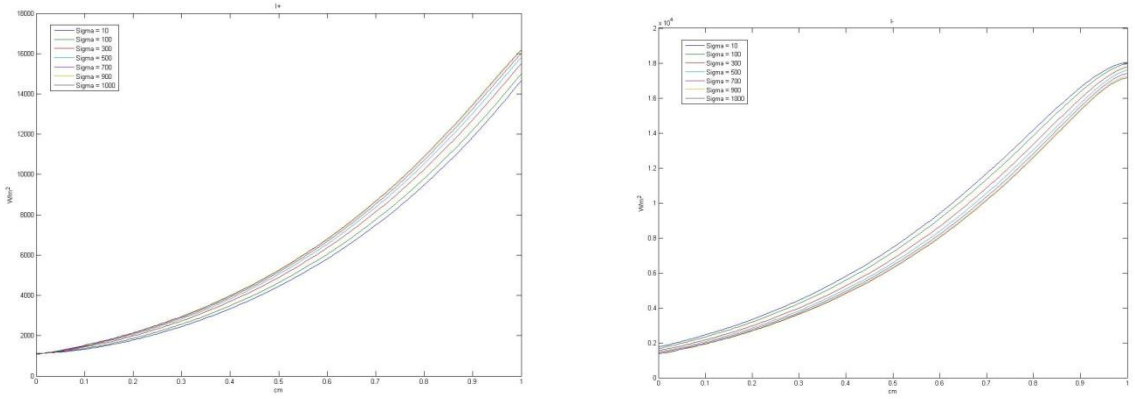


Figures 8A(Left) and 8B(Right): Comparison of Wall Heat Fluxes predicted by in-house code and FLUENT.

The most significant discrepancy between the solutions occurs at the hot wall where the in-house solver predicts the correct shape but appears to be offset from the FLUENT solution by approximately 2000 W/m^2 for all simulated values. There are two possible explanations for this discrepancy. Since all the simulations were conducted with a scattering coefficient value of $500m^{-1}$, it is possible that the optical thickness of the problem is high enough that the Schuster-Schwarzchild Approximation will naturally tend to generate erroneous estimations of intensity, which would result in inaccurate heat flux profiles. If this is the case, then solutions for heat flux at the cold wall suggest that unequal contributions of scattering and absorption to total extinction result in solution mismatch for optically thinner material but do not significantly affect solution accuracy for optically thicker media. This is counter-intuitive because the Schuster-Schwarzchild Approximation is reported to be most accurate in optically thin media with solution accuracy tending towards the analytical solution as β approaches zero. An alternative explanation has to deal with convergence of the discrete ordinates method to a mesh-independent result. Simulations for both the in-house code and FLUENT were performed such that the number of nodes between the walls in the Schuster-Schwarzchild simulations were equal to two less than the number of cells between the hot and cold walls in the FLUENT simulations. This was done so that the spatial locations of the nodes would correspond exactly to the centers of the cells where the solutions of the FLUENT equations are located. The in-house code was evaluated as having converged to a mesh independent solution because the relative error between the intensities of the two highest resolution meshes was less than 0.01. This level of convergence was not possible for the heat fluxes calculated by the discrete ordinates method used in FLUENT. The relative error between the two highest resolution meshes was 0.297 indicating that a mesh-independent solution had not been realized. If this is the case, the discrepancy is very interesting because even though the in-house solver and the FLUENT solver had the same spatial resolution, the Schuster-Schwarzchild solver was able to converge to a mesh independent solution much more rapidly than the more complicated discrete ordinates method which requires further mesh refinement to achieve mesh independence. This is the more likely source of the discrepancy between the two solutions.

The trends observed in Condition 3for intensity dependence on σ are similar to those observed in the simulations of Condition 1 where κ and σ were kept equal while β was increased. Specifically, the intensity distribution for I^+ across the slab increases in magnitude at every location across the slab as the value of scattering coefficient is increased from the minimum value of $10m^{-1}$ to the maximum

value of 1000m^{-1}, shown in Figure 9A. This increase in magnitude indicates an increase in radiation augmentation by in-scattering and can be classified as being monotonic because there does not appear to be a critical value of σ at which the I^+ intensity distribution changes shape.



Figures 9A(left) & 9B(right): Intensity profiles for Condition 3.

However, the increase in magnitude of these simulations is much less significant than those observed in Condition This result is important with regards to comparison to the I^+ distributions of Condition 2 where κ was varied because in those simulations variation in the value of κ resulted in an immediate collapse of the intensity curves to a single profile and the formation of a region near the hot wall where augmentation by volumetric emission was dominant. That behavior is not present at all in these simulations where κ is held constant and σ is varied. In fact other than the slight increase in intensity at every point along the slab, increasing the value of σ does not have any measurable effect on upper hemispherical intensity. After the value of σ is increased beyond approximately 700m^{-1}, the intensity distribution across the slab appears to be independent with respects to changes in scattering behavior.

Similar results are observed for simulated results for lower hemispherical intensity. Because of the combined effect of the boundary conditions and the linear temperature distribution that increases from the cold wall to the hot wall, the intensity distribution across the slab has upward concavity and is increasing at every location on the slab. As observed in Condition 1, the magnitude of I^- decreases as the value of extinction coefficient increases. When the value of σ is increased from 10m^{-1} to 1000m^{-1}, the magnitude of I^- decreases monotonically at every location on the slab to a profile that is independent of the value of scattering coefficient, as shown in Figure 9B. The existence of an asymptotic profile that is independent of optical thickness is a recurrent result which has appeared in all simulations in Case 2.

The simulated heat flux profiles across the slab are similar to the profiles observed in Condition 2 simulations. Everywhere in the slab, it is directed towards the cold wall and all the distributions exhibit a global maximum in the region near the hot wall. Similar to the optically thick settings simulated in Condition 2, the heat flux profiles across the slab decrease in magnitude as the value of σ is increased. The primary differences between the predicted heat flux profiles of Condition 2 and Condition 3 are the maximum values of the heat flux and the range of heat flux values at the cold wall. In these simulations where κ is fixed at 500m^{-1} and σ is varied, the maximum heat flux exhibited by any of the profiles is 15118W/m^2 at an extinction coefficient of 510m^{-1}, as shown in Figure 10. This heat flux is 37% greater than the maximum heat flux of the Condition 2 simulations, which corresponds to an extinction coefficient of 600m^{-1}. This is interesting because comparison of the intensity distributions for Condition 1 and Condition 2 suggests that radiative transfer across the slab has a greater dependence on the value of κ than on the value of σ.

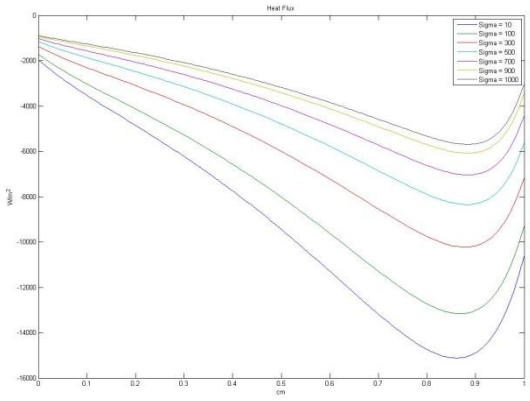
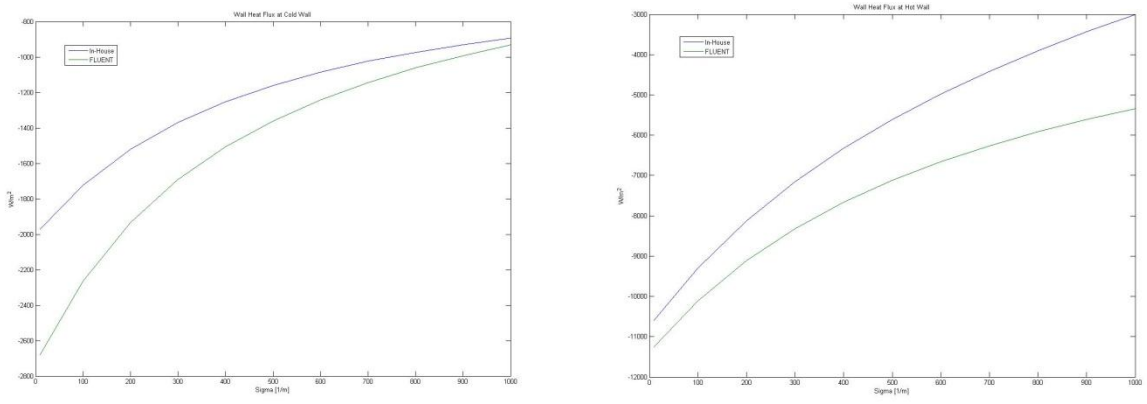


Figure 10: Heat flux profiles for Condition 3.

However, this large peak in maximum heat flux is actually due to the fact that during Condition 2 simulations the value of absorption coefficient was fixed at $500m^{-1}$. The greatest peak heat flux value in these simulations corresponds to when σ is equal to $10m^{-1}$ and κ is equal to $500m^{-1}$ when the system is dominated by absorption and emission. Therefore, the overall dependence of radiative heat flux on scattering coefficient is much less than that of absorption coefficient and the presence of peaks near the hot wall are due primarily to temperature dependent effects associated with large values for radiation augmentation by volumetric emission.

Comparison of the radiative wall heat fluxes of the Schuster-Schwarzchild solver with those generated by ANSYS FLUENT indicates that for a gray, absorbing, isotropically, scattering medium with a large absorption coefficient, the simplification of assuming constant intensities in the upper and lower hemispheres does not accurately simulate radiative transfer in a plane-parallel slab. As illustrated in Figures 11A and 11B, the Schuster-Schwarzchild Approximation significantly underestimates the values for radiative wall heat flux at both the hot wall and the cold wall. At the hot wall, the results from the in-house code exhibit a similar shape as the results from FLUENT for values of σ less that $500m^{-1}$, associated with β equal to $1000m^{-1}$. Beyond this critical value for σ, the heat fluxes calculated by FLUENT increase at a decreasing rate whereas the heat fluxes predicted by the Schuster-Schwarzchild Approximation continue to increase in a linear fashion. The results for heat flux at the cold wall exhibit the opposite behavior that what is seen at the hot wall.



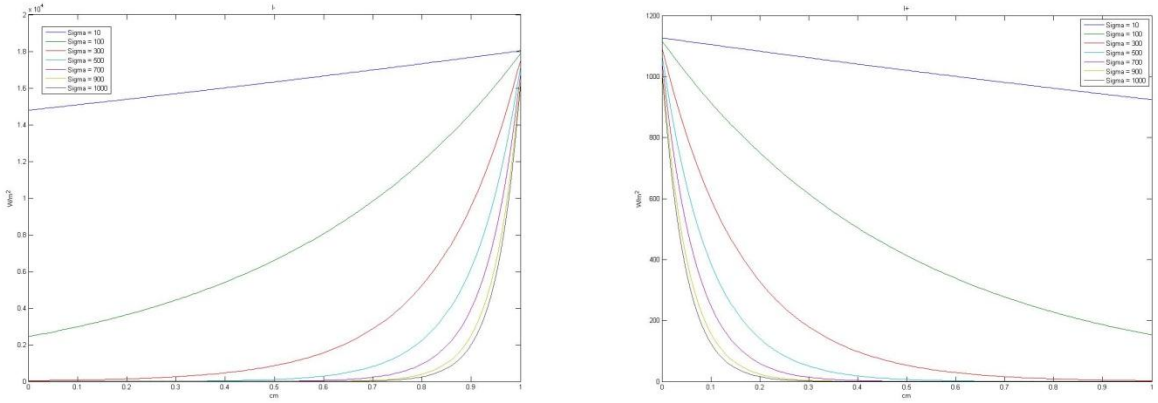
Figures 11A(Left) and 11B(Right): Wall Heat Fluxes predicted by in-house code and FLUENT for Condition 3.

Rather than the smallest discrepancy occurring when σ is smallest, the greatest error between the results of the in-house code and the FLUENT simulations occurs when σ is assigned a value of $10m^{-1}$. As the optical thickness of the slab is increased to $1500m^{-1}$, the heat fluxes predicted by the Schuster-

Schwarzchild solver converge to those predicted by FLUENT. If the results generated by FLUENT are accurate, the convergence of the cold wall heat flux at the maximum optical thickness and the convergence of hot wall heat flux at the minimum optically thickness seem to imply that the applicability of the Schuster-Schwarzchild Approximation is dependent on the temperature conditions specified in the domain and at the boundaries. However, this contradicts the results published by Xia[1] which indicate that the simplification of the RTE is most accurate for optically thin media. The most likely cause of the discrepancy between the in-house code and the FLUENT simulations is lack of convergence to a mesh-independent solution in the FLUENT simulations. As with Condition 2, the reported intensity values for the in-house code correspond to a mesh independent solution characterized by a relative error of less than 0.01 between the two finest meshes. This level of convergence was not realized for the FLUENT simulations, which implemented a full 3-dimensional discrete ordinates method which likely contributes significantly to the significant mismatch between the wall heat fluxes, especially in the optically thin regime.

CASE 3

The intensity distributions for the case of radiative transfer in a purely scattering medium with an anisotropic scattering phase function exhibit behavior that is different than those with a non-zero absorption coefficient. Specifically, since radiation augmentation from blackbody emission does not play a role, the effects of temperature that were present in the Case 1 and Case 2 simulations are absent. The lack of augmentation from blackbody emission is most apparent in the distributions of lower hemispherical intensity across the slab. As illustrated in Figure 12A, simulated profiles for nearly all values of σ decrease in a seemingly exponential manner as the distance from the cold wall increases. The only exception to this trend is the linear I^- profile associated with the scattering coefficient being equal to $10m^{-1}$, which is the optically thinnest simulated case. The presence of a linear profile in the optically thin limit is an expected result that appears in the results of Case 1 and Case 2. For the optically thinnest case, I^- decreases by a negligible amount as compared to the complete attenuation demonstrated for values of σ greater than $100m^{-1}$. However it should be noted that even though this is an expected result, the existence of a condition under which the effects of extinction, augmentation, and coupling of the transfer equations sums to a constant at every location in the slab is unexpected. With regards to the dependence of I^- on scattering coefficient, intensity decreases rapidly as σ increases.



Figures 12A(left) & 12B(right): Intensity profiles for Case 3.

The degree at which intensity decreases for large values of scattering coefficient is the most interesting feature of these simulations because unlike the results of Case 1 and Case 2, the intensity distributions do not collapse to a distinct non-zero profile in the limit of infinite optical thickness but instead rapidly

decrease to a value of zero. Similar trends are observed for distributions of upper hemispherical intensity across the slab as shown in Figure 12B. The rapid decrease seen in the intensity distributions is clear evidence of dominance of out-scattering over in-scattering in the I^+ and I^- equations of the Schuster-Schwarzchild Approximation. The only significant difference between the intensity profiles is the magnitude of values of the intensity since the values for I^- are 10^4 W/m^2 whereas those of I^+ are on the order of 10^3 W/m^2.

Schuster-Schwarzchild simulations of radiative transfer in a purely scattering plane-parallel medium with an anisotropic scattering phase function produce unrealistic results for most values of σ. As shown in Figure 13, the in-house code predicts a very large value of heat flux at hot wall which is directed towards the cold wall and decays rapidly to a value of zero at a distance of approximately 33mm into the slab. The realization of a zero heat flux value is predicted for all values of σ greater than 100m^{-1}. This result may seem plausible due to the absence of radiation augmentation due to volumetric emission. However, the only mechanism in the generalized RTE which could do this would be extinction due to absorption, which does not apply for a non-absorbing medium. Extinction due to out-scattering cannot accomplish this because out-scattering results in augmentation of intensity elsewhere in the slab resulting in a violation of the symmetry associated with the plane-parallel assumption. Therefore these results are non-physical because they do not conserve of energy. This conclusion is most apparent from comparison of the intensity profiles because the rapid decrease to zero exhibited by I^- does not account for the significant amount of energy out-scattered by I^+. In addition to this, the results for σ greater than 100m^{-1} predict net heat flux towards the hot wall at locations within 20mm of the cold wall. This is another non-physical result because energy cannot flow against a temperature gradient in the absence of work. This is a violation of entropy balance.

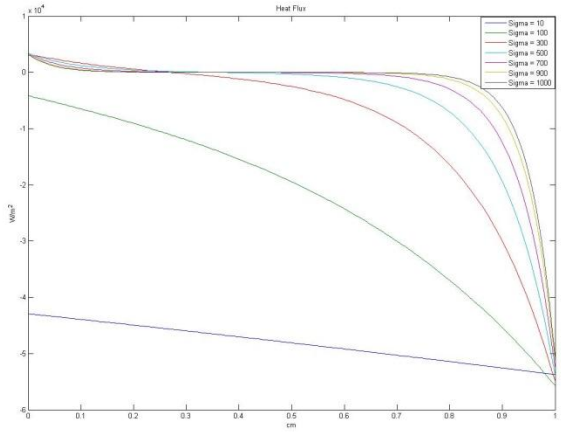
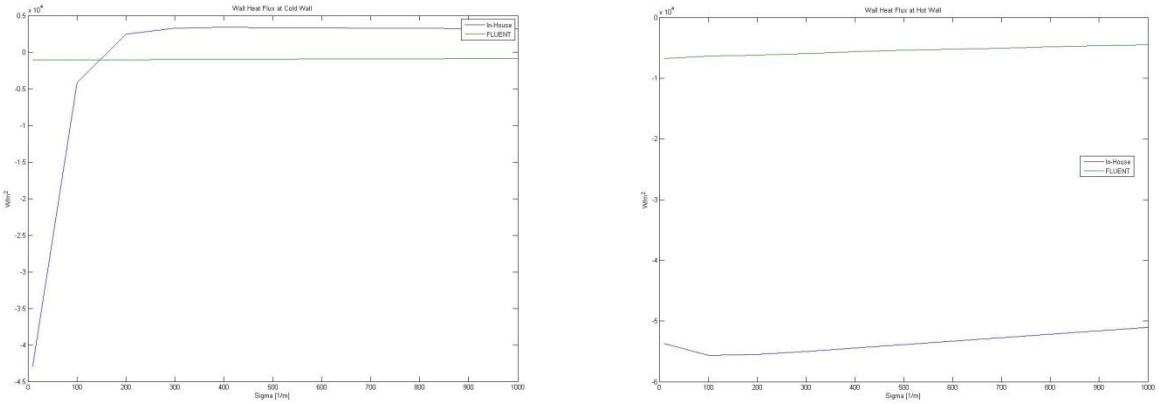


Figure 13: Heat flux profiles for Case 3.

A comparison of wall heat fluxes predicted by the in-house code to those predicted by FLUENT supports the conclusion that the in-house code fails to accurately model radiative transfer in the case of a purely scattering medium with an anisotropic scattering phase function. As demonstrated in Figures 14A and 14B, the benchmark results generated by FLUENT predict a negligible change in heat flux at the cold wall directed from the hot wall to the cold wall whereas those predicted by the in-house code drastically over-estimates the wall heat flux for the optically thinnest case and predicts reversal of the direction of heat flux for σ being greater than 100 m^{-1}. The in-house predictions for heat flux at the hot wall also demonstrate poor accuracy as they exhibit an error of 1234.6%.



Figures 14A(Left) and 14B(Right): Wall Heat Fluxes predicted by in-house code and FLUENT for Case 3.

The major conclusion of Case 3 is that the Schuster-Schwarzchild fails to accurately model radiative transfer in purely scattering plane parallel, non-absorbing medium with an anisotropic scattering phase function. It fails to do so because the simplification of the three-dimensional system to a one-dimensional system does accurately account for extinction due to out-scattering in a manner that satisfies the first and second laws of thermodynamics. This conclusion is in agreement with results published by Xia[1], which found discrepancies between an analytical solution for the Schuster-Schwarzchild equations and results of Monte-Carlo simulations when the scattering-phase function is anisotropic.

General Remarks

Implementation of the Schuster-Schwarzchild as an in-house solver for radiative transfer was very simple because it required the solution of only a single line of nodes that spanned the gap between the boundaries. Unlike a generalized three dimensional solver that would require the treatment of a multi-dimensional mesh and a sophisticated method to access all the nodes and build the coefficient matrix efficiently, the simple linear structure of the computational domain made it possible to increase the spatial resolution of the solver without significantly increasing the total simulation time, even for the implicit solver used in Case 1. This was not the case for the simulations performed using the discrete ordinates method implemented in ANSYS FLUENT. Unlike the simulations performed using the in-house solver, the discrete ordinates method took a very long time for the residuals to drop below an acceptable level. This is most likely due to a combination of the complexity of the discrete ordinates method and the fact that the FLUENT simulations were performed on a full three-dimensional mesh. With regards to comparison of the specific amount of time required to run the simulations, it took approximately 2 hours to run all the discrete ordinates simulations for Case 2 at the highest spatial resolution whereas it took less than 1 minute to run the corresponding simulations using the in-house code. In addition, the iterative explicit solver used for Case 2 and Case 3 was able to achieve mesh independence for coarser mesh resolutions than the discrete ordinates method. All reported results for in-house simulations converged to less than 5% relative error between successive meshes with some reaching convergence levels as low as 0.1%. This degree of mesh independence was not possible for all the FLUENT simulations.

APPENDIX

REFERENCES

[1]Modest, M., *Radiative Heat Transfer*, 1993, 1st ed., McGraw-Hill, New York.

[2]Xia, X., Li, D.,Sun, F., 2010, "Analytical Solution Under Two-Flux Approximation to Radiative Heat Transfer in Absorbing Emitting and Anisotropically Scattering Medium," Journal of Heat Transfer, Vol. 132, pp. 1-8

CASE 1 Code: Main.java

```
import java.io.*;
import java.lang.Math;

class Main{


        public static void main(String[] args){

                double thickness = .01;//in meters
                String filename;

                //int nodecount = 48;  //Coarse Mesh
                //int nodecount = 98;  //Normal Mesh

                //int nodecount = 152; //Fine Mesh
                int nodecount = 198; //Finest Mesh

                //-----------------------------------//-

                double wavelength;//in microns
                double kappa;
                double beta;

                //beta =  Math.pow(10,-19); //in m^-1
                //kappa = Math.pow(10,-19);//in m^-1

                //beta =  15; //in m^-1                    //for band 2
                //kappa = 15;//in m^-1                     //for band 2

                //beta =  600; //in m^-1      //for band 3
                //kappa = 600;//in m^-1                    //for band 3

                beta =  10000; //in m^-1     //for band 4
                kappa = 10000;//in m^-1    //for band 4

                //for(double i = 0.1; i<0.5;i+=0.1){
                //for(int i = 0; i<10; i++){ //for band 2
                //for(int i = 0; i<9; i++){ //for band 3
                for(int i = 5; i<550; i=i+1){ //for band 4

                //wavelength = i; //band 1
                //wavelength = 0.8+0.2*i; //band 2
                //wavelength = 2.8+0.2*i; //band 3
                wavelength = i; //band 4

                double[] mesh;

                //Spectral Intensities
                double[] IMinus;
                double[] IPlus;
                //double[] I;

                MeshGen generator;
                Solver solver = new Solver(nodecount);
                Radiation rad = new Radiation();
```

```
                Writer pen = new Writer();

                //Specify the Mesh Dimensions and Number of Computational Nodes
                generator = new MeshGen(0,thickness,nodecount);
                generator.generate();
                mesh = generator.getMesh();

                //Specify the Number of Computational Nodes
                solver = new Solver(nodecount);

                //Input Beta and the slab thickness
                solver.DiffEqConstants(beta,thickness);

                //Input the wavelength of interest and the absorption coefficient
                solver.BlackBodyConstants(wavelength,kappa);

                solver.AssembleMatrices();
                //System.out.println("");

                //System.out.println("");
                IPlus = solver.SolveIPlus();
                IMinus = solver.SolveIMinus();
                //I = solver.SolveI();


                //filename = new String(wavelength +  " Intensity");
                //pen.writeVector(I,filename);

                filename = new String(wavelength +  " I+");
                pen.writeVector(IPlus,filename);

                filename = new String(wavelength +  " I-");
                pen.writeVector(IMinus,filename);


                }//End of Wavelength Loop

                //pen.writeVector(mesh, new String("mesh"));

        }
}
```

CASE 1 Code: Solver.java

```
import java.lang.Math;
import java.io.*;

class Solver {

        final double Thot = 1000;//degrees Kelvin
        final double Tcool = 500;//degrees Kelvin

        //I+ Matrices
        public double[][] A1;
        public double[] b1;
```

```
//I- Matrices
public double[][] A2;
public double[] b2;

//I Matrices
public double[][] A3;
public double[] b3;

public double[] solution;
public int nodecount;


private double sigma,margin;
private double[] temp;

private double Aconst1;
private double Aconst2;
private double Bconst;
private double lambda;
private double T;
private double dels;

private Radiation rad;

//////////////////////////////////////////
//////////////Constructor/////////////////
//////////////////////////////////////////

public Solver(int nodes){
	margin = (double)0;
	sigma = (double)0;
	temp = new double[1];
	rad = new Radiation();
	nodecount = nodes;
}

////////////////////////////////////////////
///////////////SOLVER METHODS/////////////////
////////////////////////////////////////////

//Constants relevant for the Differencing Equations of the I+ & I- Matrices
public void DiffEqConstants(double beta, double thickness){
	dels = thickness/(nodecount+1);//divide entire thickness over discrete node spacings
	Aconst1 = 1/(4*dels);
	Aconst2 = beta;
}

//Constants relevant for the BlackBody terms in the B vector
public void BlackBodyConstants(double wavelength, double kappa){
	Bconst = kappa;
	lambda = wavelength;//In Microns
}

public void AssembleMatrices(){

	////////////////////////////////////////////////////////////////////////
	////----------Initialize and Assemble the Matrices for I+ -----------//////
```

```
////////////////////////////////////////////////////////////////////////

//-----------------A-Matrix---------------//
A1 = new double[nodecount][nodecount];

// Row[0]
A1[0][0] = Aconst2;//  beta
A1[0][1] = Aconst1;//  1/4*dels

// Row[1] to Row[nodecount-2]
for (int i = 1; i<nodecount-1; i++) {
        A1[i][i-1] = -Aconst1;// [-1/(4*dels)]
        A1[i][i] = Aconst2;//    [beta]
        A1[i][i+1] = Aconst1;//  [1/(4*dels)]
}

// Row[nodecount-1] -- First Order Backward Difference Derivative
A1[nodecount-1][nodecount-2] = -2*Aconst1;//  [-1/(2*dels)]
A1[nodecount-1][nodecount-1] = Aconst2 + 2*Aconst1;//   [beta + 1/(2*dels)]

////----------Initialize and Assemble the B-Matrix-----------//////

b1 = new double[nodecount];

T = Tcool + (Thot-Tcool)/(nodecount+2);
//System.out.print(T);

//First Element
b1[0] = rad.Planck(T,lambda,1)*Bconst/Math.PI + Aconst1*rad.Planck(Tcool,lambda,1)/Math.PI;

//Second Element to Row[nodecount-2]
for (int i = 1; i<nodecount-1; i++) {
        T = Tcool + (i+1)*(Thot-Tcool)/(nodecount+2);


        b1[i] = rad.Planck(T,lambda,1)*Bconst/Math.PI;
}

T = Tcool + (nodecount)*(Thot-Tcool)/(nodecount+2);

//Last Element
b1[nodecount-1]= rad.Planck(T,lambda,1)*Bconst/Math.PI;

////////////////////////////////////////////////////////////////////////
////----------Initialize and Assemble the Matrices for I- -----------//////
////////////////////////////////////////////////////////////////////////

//-----------------A-Matrix---------------//
A2 = new double[nodecount][nodecount];

// Row[0]
A2[0][0] = Aconst2+2*Aconst1;//  beta + 1/(2*dels)
A2[0][1] = -2*Aconst1;//  [-1/(2*dels)]

// Row[1] to Row[nodecount-2]
for (int i = 1; i<nodecount-1; i++) {
        A2[i][i-1] = Aconst1;// [1/(4*dels)]
```

```
                A2[i][i] = Aconst2;//    [beta]
                A2[i][i+1] = -Aconst1;//  [-1/(4*dels)]
            }

            // Row[nodecount-1]
            A2[nodecount-1][nodecount-2] = Aconst1;//  [1/(4*dels)]
            A2[nodecount-1][nodecount-1] = Aconst2;//   [beta]

            ////----------Initialize and Assemble the B-Matrix-----------//////

            b2 = new double[nodecount];

            T = Tcool + (Thot-Tcool)/(nodecount+2);

            //First Element
            b2[0] = rad.Planck(T,lambda,1)*Bconst/Math.PI;

            //First Element to Row[nodecount-2]
            for (int i = 1; i<nodecount-1; i++) {
                T = Tcool + (i+1)*(Thot-Tcool)/(nodecount+2);
                b2[i] = rad.Planck(T,lambda,1)*Bconst/Math.PI;
            }

            T = Tcool + (nodecount)*(Thot-Tcool)/(nodecount+2);

            //Last Element
            b2[nodecount-1] = rad.Planck(T,lambda,1)*Bconst/Math.PI +
Aconst1*rad.Planck(Thot,lambda,1)/Math.PI;

        }//End of Assemble Matrices


        /////SOLVES THE CONSTRUCTED I+ SYSTEM
        public double[] SolveIPlus(){
            double[] out = new double[nodecount];

            temp = Thomas(A1,b1);//length = nodecount

            for(int i = 0;i<nodecount;i++){
                out[i]=temp[i];
            }

            return out;
        }

        /////SOLVES THE CONSTRUCTED I- SYSTEM
        public double[] SolveIMinus(){
            double[] out = new double[nodecount];

            temp =   Thomas(A2,b2);

            for(int i = 0;i<nodecount;i++){
                out[i]=temp[i];
            }

            return out;
        }
```

```
//////////////////////////////////////////////////
//////////////////Internal Methods/////////////////
//////////////////////////////////////////////////

/*
//////----Successive Over-Relaxation Matrix Solver----/////////
public double[] SOR(double[][] AA, double[] bb, double tol, double guess, double w){
	//////Uses the Successive Over Relaxation Method to Solve the system Ax=b

	int n = AA.length;
	double[] xk_1 = new double[n];
	double[] xk = new double[n];


	//Initializes output vector with initial guess
	for (int i=0; i<n; i++) {
		xk_1[i]=guess;
		xk[i]=guess*2;
	}

	margin = 100;//arbitrarily large initial percentile error

	while (margin>tol) {//checks convergence

		for (int i =0; i<n; i++) {
			xk_1[i]=xk[i]; //assigns kth-iteration of x to xk_1 before next iteration
		}

		for (int i=0; i<n; i++) {

			sigma = 0;

			for (int j=0; j<n; j++) {

				if (j!=i){sigma = sigma + AA[i][j]*xk[j];}

			}//end of j-loop

			xk[i]=(1-w)*xk[i]+(w/AA[i][i])*(bb[i]-sigma);//updates xk with new values

		}//end of i-loop: kth iteration has completed

		margin = maxerror(xk_1,xk);

	}//end of while loop

	return xk;
}
*/

//////////////////---Thomas Algorithm---////////////////////
public double[] Thomas(double[][] AA, double[] bb){
	int n = AA.length;

	double gamma, rho;
	double[] xx = new double[n];
```

```
                ////////////////////Forward Elimination Phase//////////////////

                for (int i = 1; i < n; i++)
    {
                        gamma = AA[i][i-1]/(AA[i-1][i-1]);

                        AA[i][i] = AA[i][i]-gamma*AA[i-1][i];   //Diagonal Term
                        bb[i] = bb[i]-gamma*bb[i-1];



                }

                ////////////////////Back Substitution Phase//////////////////

                xx[n-1] = bb[n-1]/AA[n-1][n-1];

                for (int i = n-2; i>-1; i--)
                {
                        xx[i]=(bb[i]-AA[i][i+1]*xx[i+1])/AA[i][i];

                }

                return xx;
        }

}
```

CASE 1 Code: Radiation.java

```
import java.lang.Math;

class Radiation{


        public Radiation(){}

        ////Returns Black Body Emissive Power at a Given Wavelength and Temperature
        ////UNIT REQUIREMENTS: T is in Kelvin
        ////UNIT REQUIREMENTS: lambda is in microns
        public double Planck(double T, double lambda, double RI){

                double lambda_meters = lambda*Math.pow(10,-6);

                final double h = 6.626*Math.pow(10,-34); //[units] = J*s
                final double c0 = 2.998*Math.pow(10,8); //[units] = m/s
                final double k  = 1.3806*Math.pow(10,-23); //[units] = J/K

                double C1 = 2*Math.PI*h*c0*c0;
                double C2 = h*c0/(k);

                double out = (C1/(RI*RI*Math.pow(lambda_meters,5)))*(1/(Math.exp(C2/(RI*lambda_meters*T))-
1));

                //double out = (C1/(RI*RI*Math.pow(lambda,5)))*(1/(Math.exp(C2/(RI*lambda*T))-1));
```

```
                return out;
        }
}
```

CASE 2 Code: Main.java

```
import java.io.*;
import java.lang.Math;

class Main{


        public static void main(String[] args){

                double thickness = .01;//in meters

                //int nodecount = 48;  //Coarse Mesh
                //int nodecount = 98;  //Normal Mesh
                //int nodecount = 152; //Fine Mesh
                int nodecount = 198; //Finest Mesh

                //int nodecount = 1000;     //validation
                //double sigma = 500;               //validation case
                //double kappa = 500;               //validation case
                //for(double kappa = 100;kappa<2000;kappa+=100){//validation

                double kappa=100;
                double sigma=100;

                //double kappa=500;
                //double sigma=500;


                //for(double i = 100;i<1010;i+=100){

                //double kappa=i;
                //double sigma=i;

                double[] mesh;
                double[] heatflux = new double[nodecount];
                double[] divheat = new double[nodecount];//validation

                //Spectral Intensities
                double[] IMinus;
                double[] IPlus;
                double[] I;


                String filename;

                MeshGen generator;
                Solver solver = new Solver(nodecount+2);
                Writer pen = new Writer();

                //Specify the Mesh Dimensions and Number of Computational Nodes
```

```
                generator = new MeshGen(0,thickness,nodecount);
                generator.generate();
                mesh = generator.getMesh();

                //Specify the Number of Computational Nodes
                solver = new Solver(nodecount);

                //Input Beta and the slab thickness
                solver.DiffEqConstants(sigma, kappa ,thickness);

                System.out.println("Initializing I+ and I-");

                solver.InitializeIPlusIMinus(1000);//Max Iterations = 100
                //solver.AssembleMatrices();

                //System.out.println("");

                //System.out.println("");
                IPlus = solver.getIPlus();
                IMinus = solver.getIMinus();
                //I = solver.SolveI();


                 //System.out.println("\n Heat Flux \n");
                 for(int ii = 0; ii<nodecount; ii++){
                            heatflux[ii]=(Math.PI*(IPlus[ii]-IMinus[ii]));
                 }

                //System.out.println("\n Heat Flux \n");
                for(int ii = 1; ii<nodecount-2; ii++){
                           divheat[ii]=(heatflux[ii+1]-heatflux[ii-1])/(.01/(2*(nodecount+2)));
                }


                //filename = new String("Intensity -> [K,S] = " + "[" + kappa + " , " + sigma + "]" );
                //pen.writeVector(I,filename);

                filename = new String("I+ -> [K,S] = " + "[" + kappa + " , " + sigma + "]");
                pen.writeVector(IPlus,filename);

                filename = new String("I- -> [K,S] = " + "[" + kappa + " , " + sigma + "]");
                pen.writeVector(IMinus,filename);

                filename = new String("Heat Flux -> [K,S] = " + "[" + kappa + " , " + sigma + "]");
                pen.writeVector(heatflux,filename);

                filename = new String("DivHeat -> [K,S] = " + "[" + kappa + " , " + sigma + "]");
                pen.writeVector(divheat,filename);

                //pen.writeVector(mesh, new String("mesh"));
                //}
        }


}
```

CASE 2 Code: Solver.java

```
import java.lang.Math;
import java.io.*;

class Solver {

	final double Thot = 1000;//degrees Kelvin
	final double Tcool = 500;//degrees Kelvin
	final double sigma = 5.67*Math.pow(10,-8);//Stefan-Boltzman Constant Watts/(m^2*Kelvin^4)

	//I Matrices
	private double[][] A;
	private double[] b ;

	private double[] IPlus;
	private double[] IMinus;

	private double[] solution;

	private int nodecount;
	private double convergence = 0.001; // 0.1%

	private double kappa, beta, scattC;
	private double dels;


	private double Aconst1;
	private double Aconst2;
	private double Bconst1;
	private double Bconst2;

	private double lambda;
	private double T;

	//////////////////////////////////////////
	//////////////Constructor/////////////////
	//////////////////////////////////////////

	public Solver(int nodes){
		nodecount = nodes;
		IPlus = new double[nodecount];
		IMinus = new double[nodecount];
	}

	////////////////////////////////////////////
	///////////////SOLVER METHODS/////////////////
	////////////////////////////////////////////

	//Constants relevant for the Differencing Equations of the I+ & I- Matrices
	public void DiffEqConstants(double SIGMA, double KAPPA, double thickness){
		dels = thickness/(nodecount+1);//divide entire thickness over discrete node spacings
		beta = SIGMA+KAPPA;
		scattC = SIGMA;
		kappa = KAPPA;

		Aconst1 = 1/(2*dels);
```

```
		Aconst2 = beta;

		Bconst1 = kappa;
		Bconst2 = scattC/2;
	}

	//Iteratively Solves the I+ and I- Equations using an Explicit Euler Scheme
	public void InitializeIPlusIMinus(int maxIter){

		double error = 1; //Initializes Error Variable
		double temp=0;

		//Zeroth Indexed I+ Element (To right of the cool wall)
		IPlus[0]=(-2*beta*sigma*Math.pow(Tcool,4)/Math.PI +
2*kappa*sigma*Math.pow(Tcool,4)/Math.PI
					+
scattC*(sigma*Math.pow(Tcool,4)/Math.PI))*dels+sigma*Math.pow(Tcool,4)/Math.PI;

		//Last Indexed I- Element (To left of the hot wall)
		IMinus[nodecount-1]=(2*beta*sigma*Math.pow(Thot,4)/Math.PI -
2*kappa*sigma*Math.pow(Thot,4)/Math.PI
					- scattC*(sigma*Math.pow(Thot,4)/Math.PI))*(-
dels)+(sigma*Math.pow(Thot,4)/Math.PI);

		//Iterations continue until the maximum error is "converged" OR a maxIter # of iterations has passed
		int count=0;
		//System.out.println(count);
		while(count<maxIter){
			//System.out.println(count);
			//System.out.println("Error = " + error);

			if(error<=convergence){
				System.out.println(error);
				break;
			}

			error=0;//reset error to evaluate next iteration

			//System.out.println("\n IPLUS");

			//I+ Euler Scheme
			for (int i = 1;i<nodecount; i++){
				//Temperature at Node to the LEFT!!!
				//SINCE the derivative is dependent on the previous node...the temperature is
evaluated at the node to the left!!
				T = Tcool + (i)*(Thot-Tcool)/(nodecount+2);

				temp = (-2*beta*IPlus[i-1] + 2*kappa*sigma*Math.pow(T,4)/Math.PI +
scattC*(IMinus[i-1]+IPlus[i-1]))*dels + IPlus[i-1];

				if(Math.abs((temp-IPlus[i])/temp) > Math.abs(error)){
					error = Math.abs((temp-IPlus[i])/temp);
				}

				//System.out.println(temp);
				//System.out.println(IPlus[i] + "\n");
```

```
                        IPlus[i] = temp;
                        //System.out.println(IPlus[i]);
                    }

                    //System.out.println("\n IMINUS");

                    //I-Euler Scheme
                    for (int i = nodecount-2;i>-1; i--){
                        T = Tcool + (i+2)*(Thot-Tcool)/(nodecount+2);

                        // "-dels" indicates integration in the negative direction)
                        temp = (2*beta*IMinus[i+1] - 2*kappa*sigma*Math.pow(T,4)/Math.PI -
scattC*(IMinus[i+1]+IPlus[i+1]))*(-dels)+IMinus[i+1];

                        if(Math.abs((temp-IMinus[i])/temp) > Math.abs(error)){
                            error = Math.abs((temp-IMinus[i])/temp);
                        }

                        //System.out.println(temp);
                        //System.out.println(IMinus[i]);


                        IMinus[i]=temp;
                        //System.out.println(IMinus[i]);

                    }

                    count++;
                }

        }

        /////SOLVES THE CONSTRUCTED I+ SYSTEM
        public double[] getIPlus(){
            double[] out = new double[nodecount];

            for(int i = 0;i<nodecount;i++){
                out[i]=IPlus[i];
            }

            return out;
        }

        /////SOLVES THE CONSTRUCTED I- SYSTEM
        public double[] getIMinus(){
            double[] out = new double[nodecount];

            for(int i = 0;i<nodecount;i++){
                out[i]=IMinus[i];
            }

            return out;
        }


        ////////////////////////////////////////////////////
        ////////////////////Internal Methods/////////////////
```

```
	//////////////////////////////////////////////////

	//////////////////---Thomas Algorithm---//////////////////
	public double[] Thomas(double[][] AA, double[] bb){
		int n = AA.length;

		double gamma, rho;
		double[] xx = new double[n];

		/////////////////Forward Elimination Phase/////////////////

		for (int i = 1; i < n; i++)
	{
			gamma = AA[i][i-1]/(AA[i-1][i-1]);

			AA[i][i] = AA[i][i]-gamma*AA[i-1][i];   //Diagonal Term
			bb[i] = bb[i]-gamma*bb[i-1];
		}

		/////////////////Back Substitution Phase/////////////////

		xx[n-1] = bb[n-1]/AA[n-1][n-1];

		for (int i = n-2; i>-1; i--)
		{
			xx[i]=(bb[i]-AA[i][i+1]*xx[i+1])/AA[i][i];

		}
		return xx;
	}



}
```

CASE 3 Code: Main.java

```
import java.io.*;
import java.lang.Math;

class Main{


	public static void main(String[] args){

		double thickness = .01;//in meters

		//int nodecount = 48;  //Coarse Mesh
		//int nodecount = 98;  //Normal Mesh
		//int nodecount = 152; //Fine Mesh
		//int nodecount = 198; //Finer Mesh

		int nodecount = 500; //Finest Mesh

		double kappa=0;

		//double sigma=10;
```

```
for(double i = 100;i<1010;i+=100){

double sigma=i;

double[] mesh;
double[] heatflux = new double[nodecount];

//Spectral Intensities
double[] IMinus;
double[] IPlus;
double[] I;

String filename;

MeshGen generator;
Solver solver = new Solver(nodecount);
Writer pen = new Writer();

//Specify the Mesh Dimensions and Number of Computational Nodes
generator = new MeshGen(0,thickness,nodecount);
generator.generate();
mesh = generator.getMesh();

//Specify the Number of Computational Nodes
solver = new Solver(nodecount);

//Input Beta and the slab thickness
solver.DiffEqConstants(sigma, kappa ,thickness);

System.out.println("Initializing I+ and I-");

solver.InitializeIPlusIMinus(1000);//Max Iterations = 100
//solver.AssembleMatrices();

//System.out.println("");

//System.out.println("");
IPlus = solver.getIPlus();
IMinus = solver.getIMinus();
//I = solver.SolveI();


 //System.out.println("\n Heat Flux \n");
 for(int ii = 0; ii<nodecount; ii++){
            heatflux[ii]=(Math.PI*(IPlus[ii]-IMinus[ii]));
 }

double[] divheat = new double[nodecount];
//System.out.println("\n Heat Flux \n");
for(int ii = 1; ii<nodecount-2; ii++){
           divheat[ii]=(heatflux[ii+1]-heatflux[ii-1])/(.01/(2*(nodecount+2)));
}

//filename = new String("Intensity -> [S] = " +  sigma  );
//pen.writeVector(I,filename);
```

```
                filename = new String("I+ -> [S] = " + sigma );
                pen.writeVector(IPlus,filename);

                filename = new String("I- -> [S] = " + sigma );
                pen.writeVector(IMinus,filename);

                filename = new String("Heat Flux -> [S] = " + sigma );
                pen.writeVector(heatflux,filename);

                filename = new String("DivHeat -> [K,S] = " + "[" + kappa + " , " + sigma + "]");
                pen.writeVector(divheat,filename);

                //pen.writeVector(mesh, new String("mesh"));
                }
        }


}
```

CASE 3 Code: Solver.java

```
import java.lang.Math;
import java.io.*;

class Solver {

        final double Thot = 1000;//degrees Kelvin
        final double Tcool = 500;//degrees Kelvin
        final double sigma = 5.67*Math.pow(10,-8);//Stefan-Boltzman Constant Watts/(m^2*Kelvin^4)

        //I Matrices
        private double[][] A;
        private double[] b ;

        private double[] IPlus;
        private double[] IMinus;

        private double[] solution;

        private int nodecount;
        private double convergence = 0.001; // 0.1%

        private double kappa, beta, scattC;
        private double dels;


        private double Aconst1;
        private double Aconst2;
        private double Bconst1;
        private double Bconst2;

        private double lambda;
        private double T;

        ////////////////////////////////////////
```

```
	//////////////Constructor/////////////////
	//////////////////////////////////////////

	public Solver(int nodes){
		nodecount = nodes;
		IPlus = new double[nodecount];
		IMinus = new double[nodecount];
	}

	////////////////////////////////////////////
	///////////////SOLVER METHODS/////////////////
	////////////////////////////////////////////

	//Constants relevant for the Differencing Equations of the I+ & I- Matrices
	public void DiffEqConstants(double SIGMA, double KAPPA, double thickness){
		dels = thickness/(nodecount+1);//divide entire thickness over discrete node spacings
		beta = SIGMA+KAPPA;
		scattC = SIGMA;
		kappa = KAPPA;

		Aconst1 = 1/(2*dels);
		Aconst2 = beta;

		Bconst1 = 5/8*scattC;
		Bconst2 = 3/8*scattC;
	}

	//Iteratively Solves the I+ and I- Equations using an Explicit Euler Scheme
	public void InitializeIPlusIMinus(int maxIter){

		double error = 1; //Initializes Error Variable
		double temp=0;

		//Zeroth Indexed I+ Element (To right of the cool wall)
		IPlus[0]=(-2*beta*sigma*Math.pow(Tcool,4)/Math.PI +
2*Bconst1*sigma*Math.pow(Tcool,4)/Math.PI)*dels+sigma*Math.pow(Tcool,4)/Math.PI;

		//Last Indexed I- Element (To left of the hot wall)
		IMinus[nodecount-1]=(2*beta*sigma*Math.pow(Thot,4)/Math.PI -
2*Bconst2*sigma*Math.pow(Thot,4)/Math.PI)*(-dels)+(sigma*Math.pow(Thot,4)/Math.PI);

		//Iterations continue until the maximum error is "converged" OR a maxIter # of iterations has passed
		int count=0;
		//System.out.println(count);
		while(count<maxIter){
			//System.out.println(count);
			//System.out.println("Error = " + error);

			if(error<=convergence){
				System.out.println(error);
				break;
			}

			error=0;//reset error to evaluate next iteration

			//System.out.println("\n IPLUS");
```

```
                        //I+ Euler Scheme
                        for (int i = 1;i<nodecount; i++){
                                //Temperature at Node to the LEFT!!!
                                //SINCE the derivative is dependent on the previous node...the temperature is
evaluated at the node to the left!!
                                T = Tcool + (i)*(Thot-Tcool)/(nodecount+2);

                                temp = (-2*beta*IPlus[i-1] + 2*Bconst1*IPlus[i-1] + 2*Bconst2*IMinus[i-
1])*dels + IPlus[i-1];

                                if(Math.abs((temp-IPlus[i])/temp) > Math.abs(error)){
                                        error = Math.abs((temp-IPlus[i])/temp);
                                }

                                //System.out.println(temp);
                                //System.out.println(IPlus[i] + "\n");

                                IPlus[i] = temp;
                                //System.out.println(IPlus[i]);
                        }

                        //System.out.println("\n IMINUS");

                        //I-Euler Scheme
                        for (int i = nodecount-2;i>-1; i--){
                                T = Tcool + (i+2)*(Thot-Tcool)/(nodecount+2);

                                // "-dels" indicates integration in the negative direction)
                                temp = (2*beta*IMinus[i+1] - 2*Bconst1*IPlus[i+1] -
2*Bconst2*IMinus[i+1])*(-dels)+IMinus[i+1];

                                if(Math.abs((temp-IMinus[i])/temp) > Math.abs(error)){
                                        error = Math.abs((temp-IMinus[i])/temp);
                                }

                                //System.out.println(temp);
                                //System.out.println(IMinus[i]);


                                IMinus[i]=temp;
                                //System.out.println(IMinus[i]);

                        }

                        count++;
                }

        }

        public double[] getIPlus(){
                double[] out = new double[nodecount];

                for(int i = 0;i<nodecount;i++){
                        out[i]=IPlus[i];
                }
```

```
		return out;
	}

	/////SOLVES THE CONSTRUCTED I- SYSTEM
	public double[] getIMinus(){
		double[] out = new double[nodecount];

		for(int i = 0;i<nodecount;i++){
			out[i]=IMinus[i];
		}

		return out;
	}

	/*
	/////SOLVES THE CONSTRUCTED I SYSTEM
	public double[] SolveI(){
		double[] temp = new double[nodecount+2];
		double[] out = new double[nodecount+2];
		temp =   Thomas(A,b);

		out[0]=sigma*Math.pow(Tcool,4)/Math.PI;

		for(int i = 1;i<nodecount+1;i++){
			out[i]=temp[i-1];
		}

		out[nodecount+1]=sigma*Math.pow(Thot,4)/Math.PI;


		return out;

	}
	*/

	//////////////////////////////////////////////////
	//////////////////Internal Methods/////////////////
	//////////////////////////////////////////////////

	//////////////////---Thomas Algorithm---////////////////////
	public double[] Thomas(double[][] AA, double[] bb){
		int n = AA.length;

		double gamma, rho;
		double[] xx = new double[n];

		//////////////////Forward Elimination Phase/////////////////

		for (int i = 1; i < n; i++)
{
			gamma = AA[i][i-1]/(AA[i-1][i-1]);

			AA[i][i] = AA[i][i]-gamma*AA[i-1][i];   //Diagonal Term
			bb[i] = bb[i]-gamma*bb[i-1];
```

```
                }

                //////////////////Back Substitution Phase/////////////////

                xx[n-1] = bb[n-1]/AA[n-1][n-1];

                for (int i = n-2; i>-1; i--)
                {
                        xx[i]=(bb[i]-AA[i][i+1]*xx[i+1])/AA[i][i];

                }

                return xx;
        }
}
```

CASE 3 Code: scattering_phase_function.c

```
#include "udf.h"

DEFINE_SCAT_PHASE_FUNC(Anisotropic_ScattFunc,c,fsf)
{
  real phi=0;
  *fsf = 0;

        if ((c>=-1)&&((c<-0.5))){phi=1;}

        if ((c>=-0.5)&&((c<0.5))){phi=0.5;}

        if ((c>=0.5)) {phi=2;}

        return(phi);
}
```